REMARKS/ARGUMENTS

The Examiner is thanked for granting an interview on November 17, 2004. Per the agreement reached with the Examiner, the Applicant has amended the independent claims to additionally recite features recited in claims 3 and 6. It is respectfully requested that the claims 3-6 and 9-13 be cancelled without disclaimer or prejudice.

The Applicant respectfully reiterates the arguments previously submitted, and respectfully submits that the claimed invention recited in claim 1 is patentable over the cited art without these additional features. Nevertheless, solely in order to expedite execution, claims have been amended, and are now in condition for early allowance. The Applicant, however, reserves the right to pursue claims of original scope.

For the Examiner's convenience, the rejection and substance of the arguments discussed during the interview are provided below.

In the Final Office Action, the Examiner has withdrawn the objections to the specification and rejection of claims 1-4, 13-14 and 16-19 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,412,108 (*Blandy et al.).* However, the Examiner has rejected the claims 1, 7, 13 and 16 under U.S.C. §103(a) over *Blandy et al.* in view of U.S. Patent No. 4,199,811 (*Borgerson et al.)* and/or U.S. Patent No. 6,081,665 (*Nilsen et al.)* and/or U.S. publication No. 2002/0170043). This rejection is fully traversed below.

***(a) Borgerson et al.* does NOT Teach or suggest generating an internal representation of a Java macro instruction in a pair of streams that collectively represent an internal representation of a stream of Java Bytecode instructions in a Java virtual machine** (claim 1 and 16)

Claim 1 has been rejected under U.S.C. §103(a) over *Blandy et al.* in view of U.S. Patent No. 4,199,811 (*Borgerson et al.).*

In the Office Action, the Examiner has noted that *Blandy et al.* does NOT teach generating an internal representation of a Java macro instruction in a pair of streams that collectively represent an internal representation of a stream of Java Bytecode instructions in a Java virtual machine (Final Office Action, page 3). However, the Examiner has asserted that *Borgerson et al.* teaches this feature by describing that a macro instruction stream can be decomposed into four micro instruction streams, each being executed on a separate respective processor (Final Office Action, page 3, citing col. 66, lines 40-47 of Borgerson *et al.)*. Contrary to the Examiner's assertion, it is very respectively submitted that decomposing a macro instruction stream into a plurality of micro instruction streams does NOT teach or suggest generating an internal representation of a Java macro instruction in a pair of streams that collectively represent an internal representation of a stream of Java Bytecode instructions in a Java virtual machine. Clearly, the macro instruction of *Borgerson et al.* is NOT a Java macro instruction that represents two or more Java Bytecodes. It should also be evident that the plurality of micro instruction streams of *Borgerson et al.* are NOT internally represented in a virtual machine. Instead, the micro instructions of *Borgerson et al.* correspond to a local processor, preferably utilizing micro processor LSI integrated circuit (*Borgerson et al.*, Abstract).

Moreover, it is respectfully submitted that *Borgerson et al.* does NOT teach or suggest this features because, among other things, *Borgerson et al.* does not even pertain to a virtual machine. Instead, *Borgerson et al.* pertains to a system for concurrently and simultaneously performing a plurality of micro instructions in performance of a single macro instruction stream (*Borgerson et al.*, Summary). As such, *Borgerson et al.* cannot possibly teach or suggest internal representation of a Java Macro instruction in a virtual machine.

Furthermore, it is respectfully submitted that Examiner has failed to establish a prima facie of obviousness because the Examiner has asserted that the combination of the cited references allows multiple processors to process a macro instruction (Final Office Action, page 3). It is respectfully submitted that the mere assertion that a combination is useful does no establish a prima facie of obviousness. The Examiner needs to provide a suggestion in the cited art for combining the cited references. Moreover, as suggested above, the system of *Borgerson et al.* cannot be combined with the virtual machine of *Borgerson et al.*

***(b) Nilsen et al.* does NOT teach or suggest the claimed method of generating a Java macro instruction during the Bytecode verification**

In the Final Office Action, the Examiner has noted that *Blandy et al.* does NOT teach that features: (a) counting, (b) determining, and (c) generating occur during Bytecode verification (Final office Action, page 5-6). However, the Examiner has asserted that *Nilsen et al.* teaches this feature by stating that inlining optimizations can be performed during Bytecode verification through a ROMizer tool (Final office Action, page 5, citing *Nilsen et al.* col. 63, line 64 to col. 64, line 5). Initially, it is respectfully submitted that this assertion does not address the recited combination of (a) counting, (b) determining, and (c) generating features.

It is noted that the ROMizer tool analyzes and verifies byte code and performs byte-code and constant-pool transformations described. Additionally, the ROMizer tool supports standard compiler transformations designed to optimize the performance of executed code. These optimizations include in-lining of small and/or performance critical methods, relocation of loop-invariant code outside the loop, and constant folding (*Nilsen et al.* col. 63, line 64 to col. 64, line 5).

Contrary to the Examiner's assertion, however, it is respectfully submitted that the ROMizer tool of *Nilsen et al.* does NOT teach that optimization can be performed during Bytecode verification. ROMizer tool of *Nilsen et al.* verifies Bytecode and also supports standard compiler transformation designed to optimize performance of executed code, but this does NOT mean that ROMizer tool of *Nilsen et al.* teaches that optimization is performed during Bytecode verification.

Moreover, it is respectfully submitted that ROMizer tool of *Nilsen et al.* does NOT teach or suggest that the combination of the features: (a) counting, (b) determining, and (c) generating are performed during Bytecode verification.

Again, it is noted that *Blandy et al.* teaches reading bytecode instructions during Java Bytecode verification. *Blandy et al.,* however, teaches optimization of the method after the Bytecode verification process (i.e., after class loading and linking), but prior to initialization and execution of the method (*Blandy et al.*, col. 4, lines 63-67). Thus, *Blandy et al.* teaches away from performing these operations during Bytecode verification because it teaches first loading and verifying the method before analyzing it for optimization (*Blandy et al.,* col. 2, lines 40-44, col. 4, lines 63-67). Accordingly, it is respectfully submitted that *Blandy et al.* cannot be combined with another reference to teach or suggest the claimed invention because performing (a) counting, (b) determining, and (c) generating features during Bytecode verification is contrary to the methodology of *Blandy et al.* which teaches optimization of a method after the Bytecode verification process.

(c) The cited art does NOT teach or suggest a pair of streams including a code stream that is designated to store code and a data stream that is desinated to store data inside a virtual machine

Clearly, this feature is NOT taught or suggested by *Borgerson et al.* Again, it should also be noted that *Tommesan*i illustrates a programming model invented by John von Neuman in the 1940's where, at each processing step, a control unit emits one instruction that operates on a datum (data) obtained from a memory unit (*Tommesan*i, text and accompanying figure on page 1). However, it is very respectfully submitted that the SISD computer does not teach generating an internal representation of a Java macro instruction in a pair of streams in a Java virtual machine.

CONCLUSION

Based on the foregoing, it is submitted that all pending claims are patentably distinct over the cited art of record. Additional limitations recited in the independent claims or the dependent claims are not further discussed because the limitations discussed above are sufficient to distinguish the claimed invention from the cited art. Accordingly, Applicant believes that all pending claims are allowable and respectfully requests a Notice of Allowance for this application from the Examiner.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 500388 (Order No. SUN1P828). Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

[illegible]

R. Mahboubian
Reg. No. 44,890

P.O. Box 778
Berkeley, CA 94704-0778
(650) 961-8300